Micro-Processor and Embedded Systems Lab-Session 2 Report for Instruction Memory and Data Memory

First Name : Yagna Srinivasa Harsha

Last Name : Annadata

Net ID : yxa210024

UTD ID : 2021641648

Email Id : yxa210024@utdallas.edu

Date : 2nd September 2022.

This week we were asked to create a Instruction and Data Memory modules. I was able to successfully create them.

Instruction Memory

The Instruction Memory (IM) stores all the prefetch instructions. It is composed of 5 major components: the PC (Program Counter) Unit, PC Decoder, INBUF (an input buffer to the IM), IM Storage, OUTBUF (an output buffer to the internal bus)

The output from the PC Decoder asserts one of the memory locations' READ or WRITE signal. When the WRITE signal of an IM Memory location is asserted, the value in the INBUF is written into that memory location. On the other hand, when the READ signal of an IM Memory location is asserted, the value from that storage location is read to the OUTBUF

Data Memory

The Data Memory (DM) stores all the variables.

The output from the PC Decoder asserts one of the memory locations' READ or WRITE signal. When the WRITE signal of an DM Memory location is asserted, the variable or the data in the INBUF is written into that memory location. On the other hand, when the READ signal of an IM Memory location is asserted, the data or variable from that storage location is read to the OUTBUF.

Data bytes are arbitrary bits of information to be processed by the computer. Instructions are special sequences of bytes that cause the microprocessor to perform specific tasks.

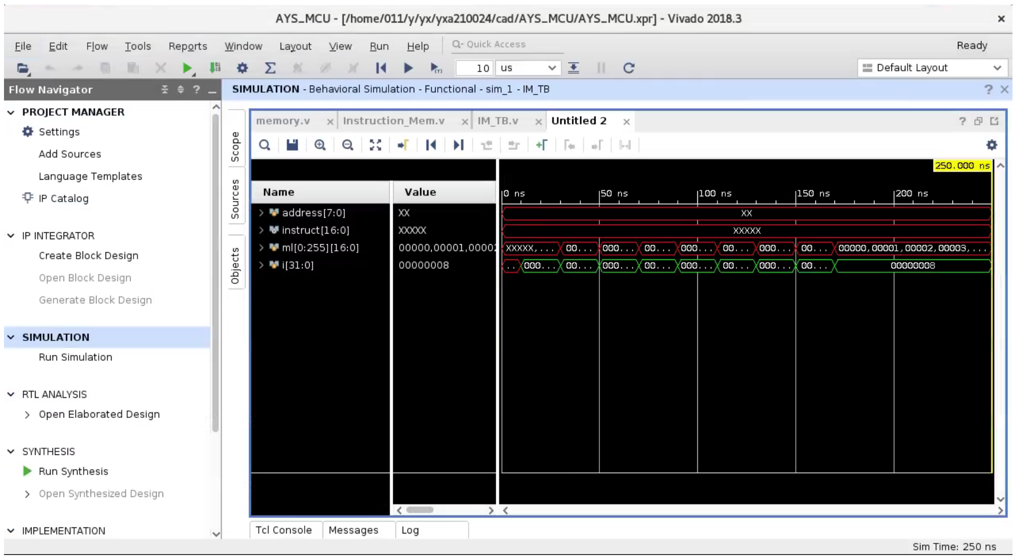
```
********************Instruction Memory***************************
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/08/2022 07:20:20 PM
// Design Name:
// Module Name: Instruction_Mem
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Instruction_Mem(

   input      [7:0]  address,
                 output wire [16:0] instruct );

  integer i;
```

```
            reg [16:0] ml[0:255];
            initial
            begin
   for(i=0;i<255;i=i+1)
       begin
          ml[i]<=0;
       end
 end
always @(address);
assign instruct=ml[address];
endmodule
```

```
*******************Instruction Memory Test Bench**************************
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/08/2022 07:25:05 PM
// Design Name:
// Module Name: IM_TB
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////



module IM_TB();

reg [7:0] address;

wire [16:0] instruct;

reg [16:0] ml[0:255];

integer i;


Instruction_Mem IM(address,instruct);

initial begin


#10

for(i=0;i<=7;i=i+1)

begin

   #20

   ml[i]=i;

end

end

initial

begin

#250 $stop;

end

endmodule
```

********************Instruction Memory Simulation***************************



SIMULATION - Behavioral Simulation - Functional - sim_1 - IM_TB

Name	Value
address[7:0]	XX
instruct[16:0]	XXXXX
ml[0:255][16:0]	00000,00001,00002
i[31:0]	00000008

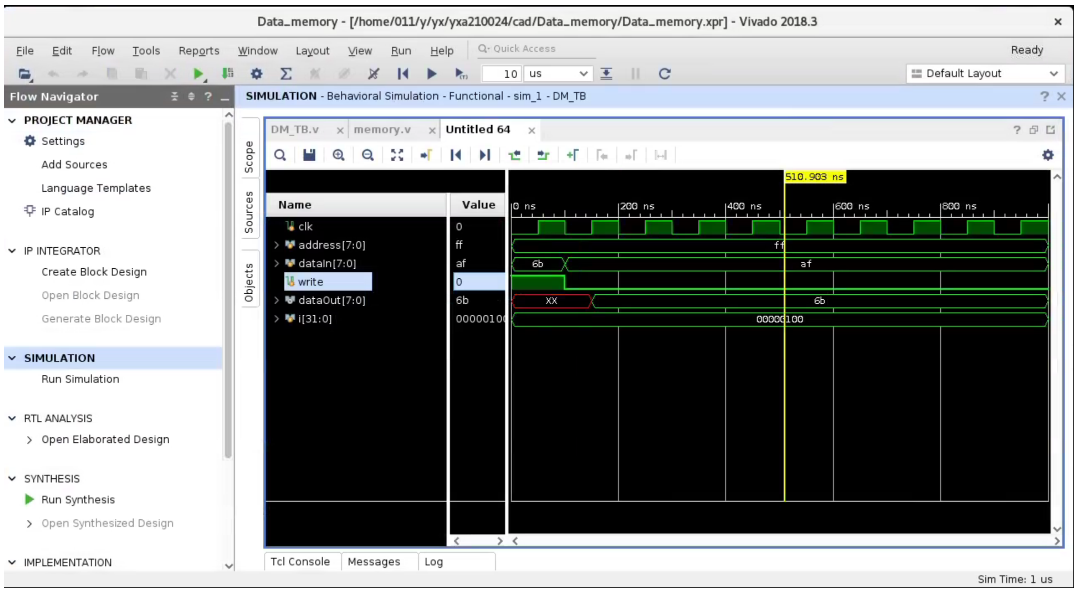
```
0 ns      50 ns      100 ns      150 ns      200 ns      250.000 ns
address:  XX
instruct: XXXXX
ml:       XXXXX,... 00... 000... 00... 000... 00... 000... 00... 00000,00001,00002,00003,...
i:        .. 000... 00... 000... 00... 000... 00... 000... 00... 00000008
```

Sim Time: 250 ns

```
******************Data Memory************************
`timescale 1ns / 1ps


module Data_mem( input          clk,

                input      [7:0] address,

                input          write,

                input      [7:0] dataIn,

                output reg[7:0]mem[255:0],

                output wire [7:0] dataOut );

  reg[8:0] addr_reg;

  always @(posedge clk ) begin

     if(write)

        mem[address] = dataIn;

                         else

                                  addr_reg <= address;



  end
```

```
        assign dataOut = mem[address];
endmodule


*******************Data Memory Test Bench**************************
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/08/2022 07:52:57 PM
// Design Name:
// Module Name: DM_TB
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////



module DM_TB();


reg clk;
reg     [7:0] address;

reg     [7:0] dataIn;

reg     write;

wire     [7:0]mem[0:255];

wire [7:0] dataOut;


reg[8:0] addr_reg;


Data_mem DM(clk,address,write,datain,mem,dataOut,addr_reg);

initial

begin

clk=0;

write=1;

end

always #50 clk=~clk;

endmodule
```

*******************Data Memory Simulation*************************


Data_memory - [/home/011/y/yx/yxa210024/cad/Data_memory/Data_memory.xpr] - Vivado 2018.3

SIMULATION - Behavioral Simulation - Functional - sim_1 - DM_TB

Name	Value
clk	0
address[7:0]	ff
dataIn[7:0]	af
write	0
dataOut[7:0]	6b
i[31:0]	00000100

Sim Time: 1 us